الباب الثاني
العمليات (Processes)

إدارة العمليات (process management) هو جزء هام من نظام التشغيل ويعتني بكل ما يتعلق **بالعمليات من**:

- **مفهوم العمليات (processes)**
- الخيوط أو العمليات الخفيفة
- **جدولة المعالج (CPU scheduling)**
- **تزامن العمليات (Process synchronization)**
- **الاختناق او الايصاد بين العمليات (Deadlocks)**

سنتحدث في هذا الباب عن العمليات.

2.1 مقدمة

قديما كانت نظم التشغيل تسمح فقط بتشغيل **برنامج واحد في اللحظة الواحدة**، هذا البرنامج يتحكم ويستأثر بكل موارد الحاسب من معالج وذاكرة وأجهزة دخل وخرج وملفات،... وغيرها. الآن **أصبحت انظمة التشغيل الحديثة تسمح لأكثر من برامج بأن يعمل في وقت واحد**. متشارك بذلك في الموارد مما أسهم بشكل فعال في تحسين أداء الحاسب وزيادة إنتاجيته (Throughput).

أيضا الإدارة الجيدة لموارد الحاسوب من قبل نظام التشغيل تؤثر تأثيرا مباشرا على الأداء. أهم موارد الحاسوب هو المعالج الذي يقوم بتنفيذ برامجنا.
البرامج قد تكون برامج نظام التشغيل، المترجمات، الأوفيس، الألعاب، وبرامج المستخدم الأخرى.
يتحول البرنامج إلى عملية عند ما نقوم بتشغيله.

2.2 مفهوم العملية (Process Concept)

البرنامج يكون في **شكل ملف** عندما يكون مخزن **بالقرص الصلب** (أو أي وسيط تخزين ثانوي) وعندما ننقر عليه نقرا مزدوجا فإننا نطلب من نظام التشغيل تنفيذه، فيقوم نظام التشغيل **بتحميله من القرص** الصلب (أو أي وسيط تخزين) الموجود به، او الفلاش أو الأسطوانة **(إلى الذاكرة الرئيسية (الرام)** ليبدأ التنفيذ، **هنا يتغير اسم البرنامج من ملف إلى عملية (Process).**

2.3 تعريف العملية

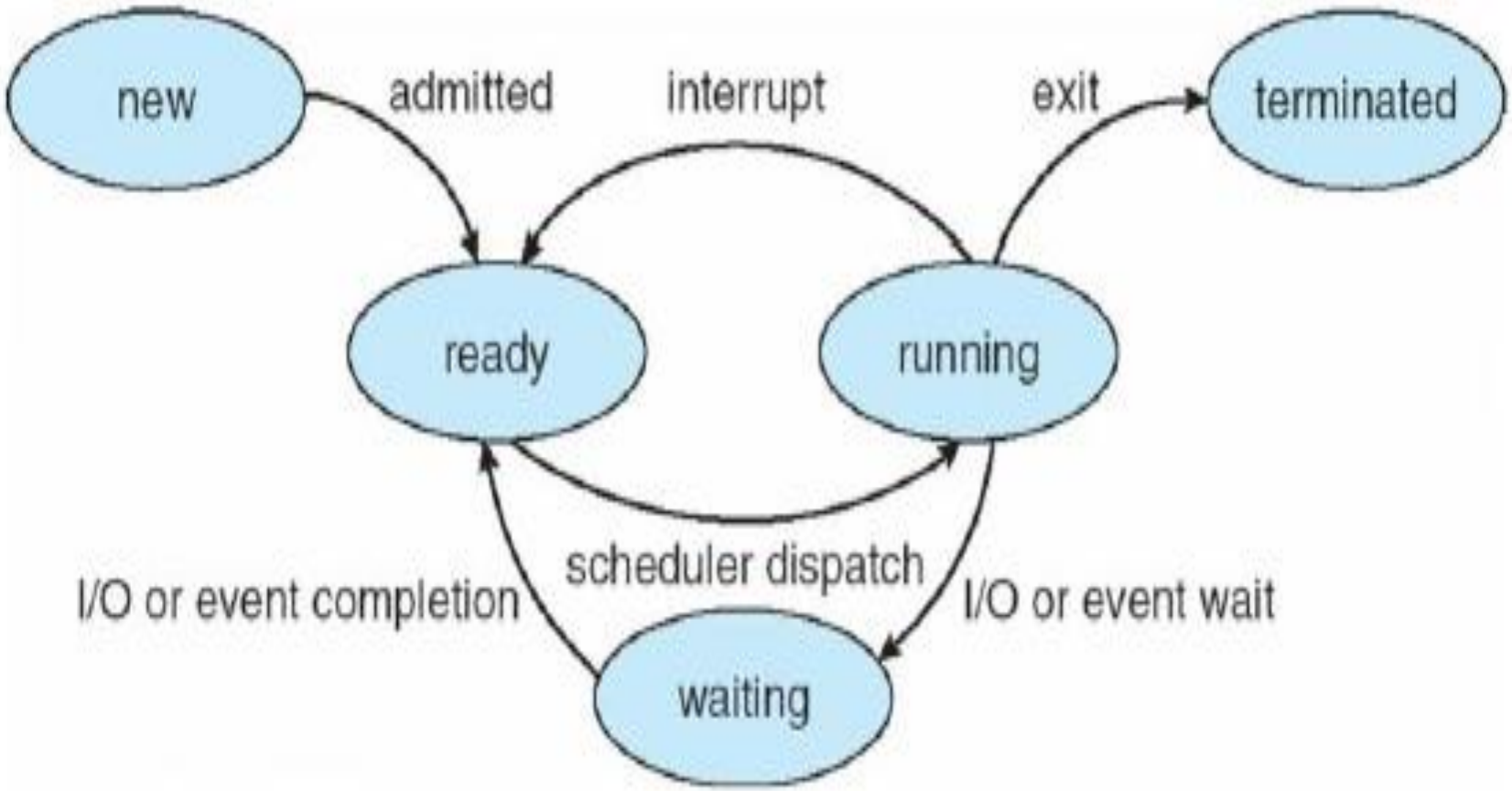
- العملية هي **برنامج شغال (تحت التنفيذ)**، أحيانا نطلق عليها **عمل (job)** أو **مهمة (task).**
- هي حالة **البرنامج اثناء التشغيل**.
- **الوحدة** التي يمكن **تخصيصها او تنفيذها** من قبل **المعالج**.
- هي **وحدة النشاط** التي تتميز **بتنفيذ سلسلة من التعليمات**، بما في ذلك بيانات الحالة الحالية والموارد المستخدمة.

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources

2.4 حالات العملية (Process states)

تحميل البرامج في الذاكرة يجعل هذه البرامج **جاهزة للتنفيذ (ready)**، عند بداية تنفيذ البرنامج داخل المعالج يصبح **شغال running**، قد يستمر المعالج في تنفيذ البرنامج حتى يكتمل، وقد **يوقف المعالج** البرنامج الشغال مؤقتا لسبب ما، فيصبح البرنامج في هذه الحالة **محجوز blocked**، وقد يشتغل برنامج آخر أكثر أهمية (مثلا). إذاً **تحميل البرنامج بالذاكرة يسمى عملية**، هذه العملية **يتغير وضعها من حال إلى حال**، **كما موضح (الشكل 2-1):**

- **جديد (New)** : العملية تم إنشاءها **وجاهزة للتحميل**.
- **حالة الجاهزية (ready state)** : العملية **تم تحميلها في الذاكرة** وأصبحت **جاهزة للتنفيذ.**
- **حالة التنفيذ (running state):** العملية **بدأت التنفيذ داخل المعالج** (يتابع مسجل عداد البرامج تسلسل تنفيذ أوامر العملية).
- **حالة الحجز أو الانتظار (blocked state or waiting state):** عندما يوقف المعالج عملية، تصبح هذه العملية محجوزة. يتم توقيف العملية لأسباب عدة مثل الحاجة لتشغيل عملية أخرى أكثر أهمية (في مثل هذه الحالة عادة يتم وضعها في **حالة الجاهزية**)، أو أن العملية تنتظر حدث ما لم يتم بعد **(even)**، أو أن الزمن الذي خصص للعملية قد اكتمل **(المشاركة الزمنية).**
- **الانتهاء (terminated):** هنا تكون العملية قد انتهى عملها، فتقوم **بإخلاء طرفها** و **تحرير الموارد** التي كانت تستخدمها، **وإخلاء الذاكرة** التي كانت تحتجزها قبل الخروج.



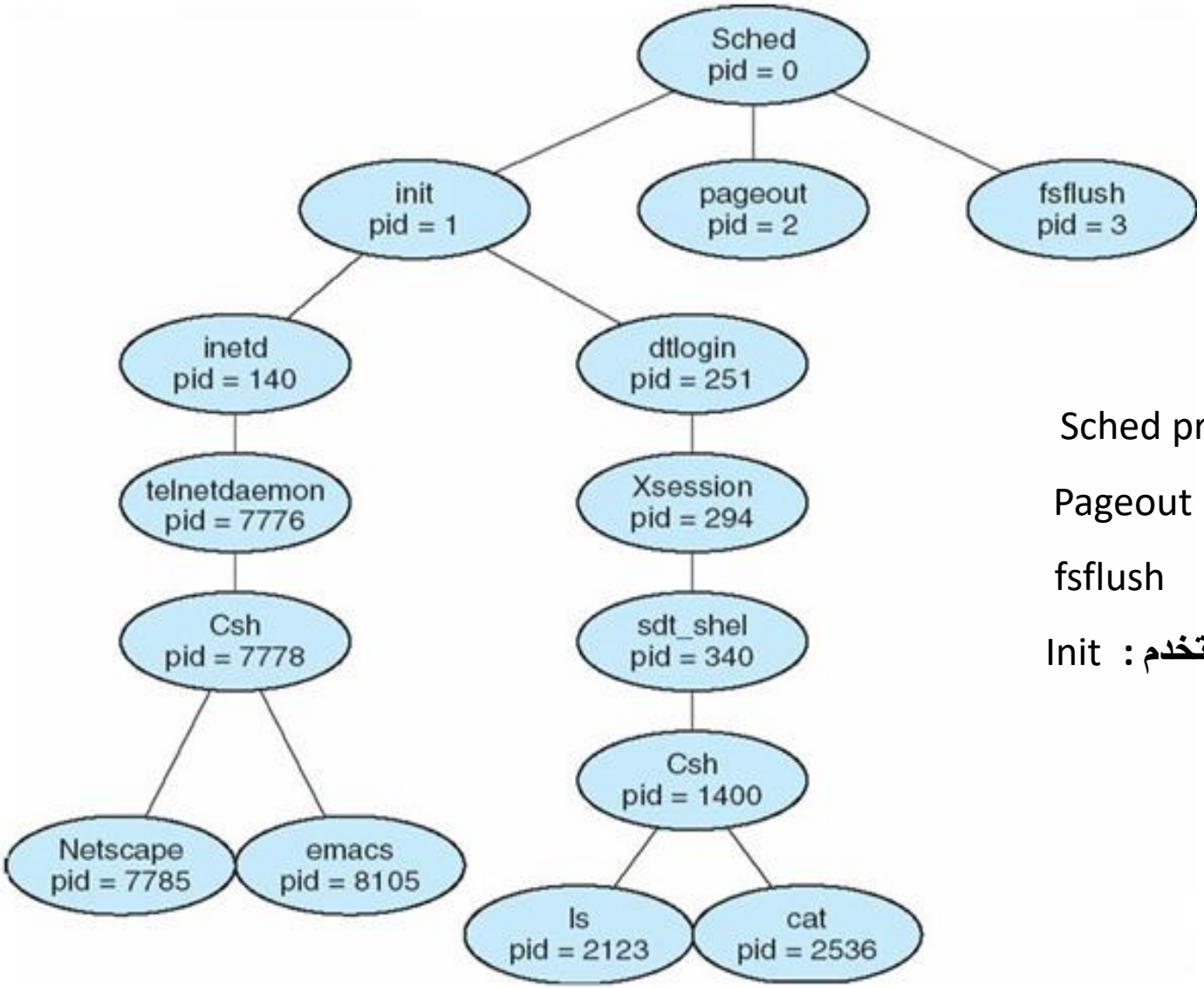
شكل رقم (2-1): حالات العملية.

2.5 إنشاء العملية (Process creation)

عند إنشاء العملية **تُعرف وتدار برقم غير متكرر** يسمى **رقم تعريف العملية (Process IdentificationNumber (PID).**

هنالك أسباب مختلفة لإنشاء العملية مثل:

- **تهيئة النظام:** عند إقلاع نظام التشغيل تنشأ العديد من العمليات، منها ما **يعمل في الخلفية background**، ومنها ما يعمل **ويتخاطب مع المستخدم**.
- **عملية تتطلب إنشاء عملية أخرى:** أحيانا تقوم عملية منفذة بتشغيل إنشاء (عملية أخرى) تساعدها في عملها، تعتبر العملية **الأولى العملية الأب (parent)** للعملية الثانية والتي تعتبر **الثانية العملية الابن child**، العملية الابن يمكنها إنشاء عمليات اخرى (قد تعمل هذه العمليات - أبناء لها مما قد يكون شجرة من العمليات) معا في وقت واحد أو تنتظر بعضها البعض (الشكل 2-2).
- **طلب المستخدم إنشاء عملية جديدة:** عند ما **ينقر المستخدم نقرا مزدوجا على أيقونة برنامج** فهذا طلب من المستخدم لإنشاء عملية جديدة.
- **المهام المحزمة (batch) :** هنا يضع المستخدم **حزمة من العمليات ويطلب من نظام التشغيل تنفيذها**، فيقوم النظام **بتنفيذ العملية الأولى في الحزمة**، ثم متى ما أتيحت له موارد العملية الثانية سيقوم بإنشائها وتنفيذها وهكذا إلى أن ينفذ كل العمليات الموجودة بالحزمة.



- **الجدولة :** Sched process pid=0
- **عملية ادارة الذاكرة :** Pageout
- **عملية ادارة الملفات :** fsflush
- **الاب لجميع عمليات المستخدم :** Init

شكل رقم (2-2): الشجرة المتكونة نتيجة تنفيذ عملية ما

- **إنشاء عملية جديدة على لينكس (fork())**

يمكن استخدام استدعاء النظام **fork()** لإنشاء عملية جديدة، وهي **لا تحتاج مدخلات arguements**، وعند استدعاءها **ترجع لنا برقم العملية التي أنشأها (process ID)** إذاً **هدف fork() هو إنشاء عملية جديدة تكون أبن للعملية التي استدعتها**. بعد أن يتم إنشاء العملية الأبن، تنفذ العملية الابن والعملية الأب الأمر الذي يلي fork() . لذلك لابد من **التمييز** بين العملية الابن والعملية الأب وذلك باختبار **القيمة الراجعة** من **fork()** :

- فإذا كانت القيمة الراجعة من fork() **سالبة**، فهذا يعني أن إنشاء العملية قد **فشل**.
- إذا كانت القيمة العائدة من الدالة fork() **صفر للعملية الابن**، فهذا يعني أن العملية قد **تم إنشاءها بنجاح**.
- ترجع fork() **رقم موجب للعملية الأب** يمثل رقم العملية (**process ID**)
- رقم العملية هو **متغير من النوع pid_t** المعرف في . **sys/types.h** ويمكن تمثيل العملية برقم، ويمكننا **استخدام الأمر getpid() للحصول على رقم العملية**.

- **البرنامج التالي** يوضح كيفية استخدام **()fork**، حيث قمنا بإنشاء عملية بإستدعاء **()fork**، ثم حصلنا على رقم تعريف العملية بالأمر **()getpid** ، أمر **printf** سينفذ مرة بواسطة العملية الأب ومرة بواسطة العملية الإبن، مخرجا **قيمتين مختلفتين للمتغير pid**. يمكن اختصار خطوتي إنشاء العملية والحصول على رقم العملية **pid=getpid()** ; **fork()** في أمر واحد هو **pid=fork()**.

يمكن استخدام استدعاءات نظام أخرى في لينكس مثل استدعاء النظام **(exec)** للتنفيذ، **(exit)**لإنهاء العملية.

```
#include <stdio.h>
#include <sys/types.h>
int main(void) {
   pid_t pid;  ⟵ متغيرات
   fork();  ⟵ استدعاء / نداء
   pid = getpid();  ⟵ الحصول على رقم العملية
   printf("This line is from pid %d, value = %d\n", pid, i);  ⟵ تنفيذ الاب/الابن
}
```

Q- Calculate number of times **hello** is printed

```
#include <stdio.h>
#include <sys/types.h>
int main()
{
        fork();  ⟶ 1
        fork();  ⟶ 2
        fork();  ⟶ 4
        printf("hello\n");
        return 0;
}
```

Answer = 8

hello
hello
hello
hello
hello
hello
hello
hello

2.6 إنهاء العملية (Terminate Process)

بعد تنفيذ العملية لآخر أمر فيها ستطلب من نظام التشغيل أن يقوم **بحذفها وذلك باستدعاء نداء النظام exit** مثلا. **مخرجات** العملية المحذوفة **ترسل للعملية الأب عبر استدعاء النظام للنداء wait**، بينما يقوم نظام التشغيل **بتحرير كل موارد العملية** .

أسباب انتهاء العملية:

- انتهاء **طبيعي** (اكتمل عملها).
- انتهاء **الزمن الكلي المخصص** لهذه العملية.
- **عدم توفر مساحة** من الذاكرة.
- **انتهاك حدود معينة** غير مسموح للعملية بالوصول اليها (bound violation).
- محاولة **استخدام مورد غير مسموح** باستخدامه –ملف مثلا (protection error).
- خطأ حسابي مثلا **القسمة على 0** (Arithmetic error).
- العملية انتظرت اكثر من **الزمن المخصص** (Time overrun).
- حدوث **خطأ اثناء عملية دخل/خرج** (I/O failure).
- محاولة **تنفيذ تعليمة بطريق الخطأ** – مثل الى مكان يحتوي **بيانات** وتنفيذها على اساس **تعليمة** (Invalid instruction).
- محاولة تنفيذ **تعليمات خاصة بنظام التشغيل** (Privileged instructions).
- **سوء استخدام البيانات** – عدم اعطاء **قيمة للمتغير او اختلاف النوع** (Data misuse).
- الانهاء نتيجة لحدوث **الايصاد** من قبل المشغل او نظام التشغيل (Operator or OS).
- تم **إنهاءها بعملية أخرى**

قد يقوم الأب بإنهاء العملية الابن (abort) اذا:

- زاد الابن في عدد الموارد المخصصة له.
- لم يعد الأب يحتاج لما يقوم به الابن (المهمة المنفذة لم نعد بحاجة لها).
- إذا أنهى الأب عمله **exiting** بعض نظم التشغيل لا تسمح للأبن بمواصلة التنفيذ إذا أنهى الأب عمله.
- كل الأبناء سينتهون بانتهاء الأب **cascading termination .**

يمكن تطبيق استدعاءات النظام التي تنشئ أو تنهي عملية معينة في **نظام التشغيل لينكس** من داخل **برنامج C**

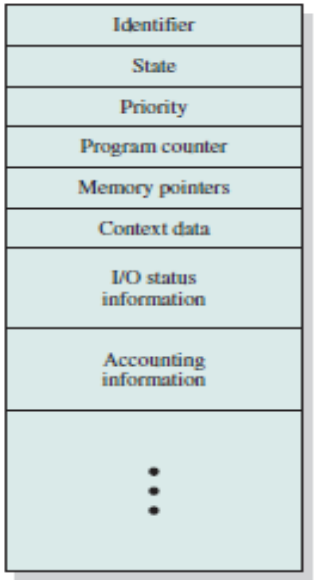
أمثلة لاستدعاءات النظام الموجودة في لينكس:

- fork() لإنشاء عملية جديدة.
- exec() لتنفيذ عملية.
- exit() لإنهاء عملية والخروج.
- wait() للانتظار.
- Kill() لإنهاء عملية.

2.7 معلومات العملية *Process Control Blocks (PCB)*

لكل **عملية بنية بيانات data structure** تسمى (**PCB**) تخزن فيها **المعلومات الأساسية للعملية** (شكل رقم 2.4)

- رقم تعريف العملية (process identification) **pid**
- حالة العملية (process state) **PS**
- محتوى عداد البرامج (Program counter) **PC**
- بيانات السياق : تشمل مسجلات المعالج CPU registers
- معلومات إدارة الذاكرة . Memory-management information
- معلومات الحسابات . Accounting information
- معلومات حالات الدخل والخرج . I/O status information
- مقدار ما نفذ من العملية.
- مكان الذاكرة المستخدم من قبل العملية.
- الموارد التي تستخدمها العملية مثل الملفات المفتوحة بواسطة العملية.
- أولوية العملية.



مؤشر

حالة العملية
رقم العملية
عداد البرامج
محتوى المسجلات
حدود الذاكرة
الملفات المفتوحة
.....

شكل رقم (2-4): بينية معلومات العملية (PCB).

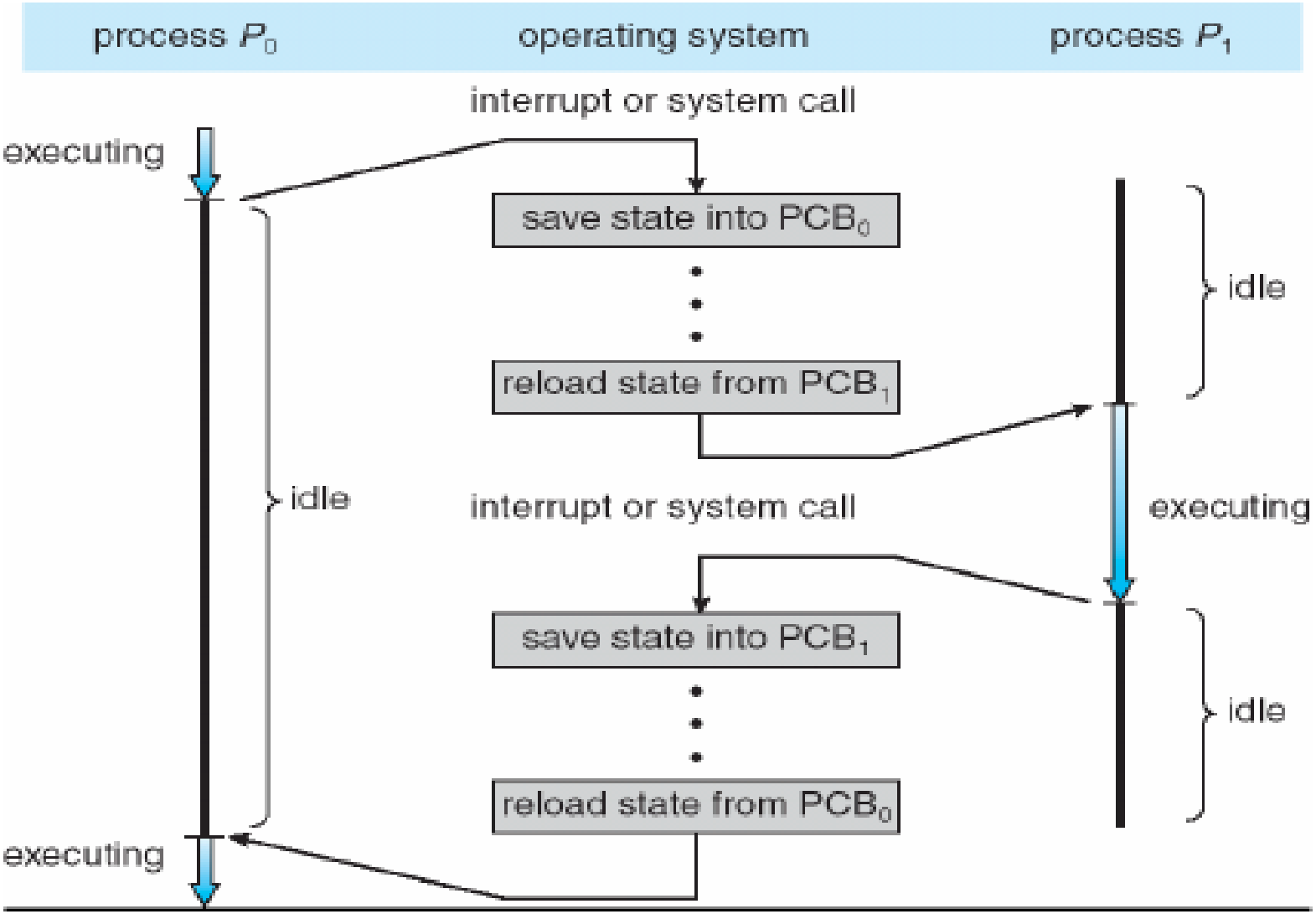
- **Identifier:** A unique identifier associated with this process, to distinguish it from all other processes.
- **State:** If the process is currently executing, it is in the running state.
- **Priority:** Priority level relative to other processes.
- **Program counter:** The address of the next instruction in the program to be executed.
- **Memory pointers:** Includes pointers to the program code and data associated with this process, plus any memory blocks shared with other processes.
- **Context data:** These are data that are present in registers in the processor while the process is executing.
- **I/O status information:** Includes outstanding I/O requests, I/O devices (e.g., disk drives) assigned to this process, a list of files in use by the process, and so on.
- **Accounting information:** May include the amount of processor time and clock time used, time limits, account numbers, and so on.

2.7 تغيير السياق (*Context switch*)

تستخدم **معلومات العمليات عندما تتحول العملية من حالة التنفيذ إلى حالة الجاهزية أو الحجز**. عندها يقوم المعالج **بتوقيف عملية وتنفيذ عملية أخرى**، حيث يتم **تعديل وحفظ معلومات العملية التي تم توقيفها** (مثلا العملية **P0** في **PCB0**، ثم يتم تحميل معلومات العملية المراد تنفيذها ، مثلا **P1** من **PCB1**

إذا أراد نظام التشغيل تنفيذ **P0** مرة أخرى فسيقوم بتعديل وحفظ معلومات **P1** في **PCB1** ثم تحميل معلومات **P0** من **PCB0** حيث يستطيع مواصلة التنفيذ من آخر نقطة وقفت فيها العملية **P0** (الشكل 2-5)

الزمن المستغرق في الانتقال بين عمليتين يكون مهدور وغير مستفاد منه (**Overhead**) لكنه لا مناص منه، حيث لا يقوم النظام بأداء أي عمل مفيد في هذه الفترة.



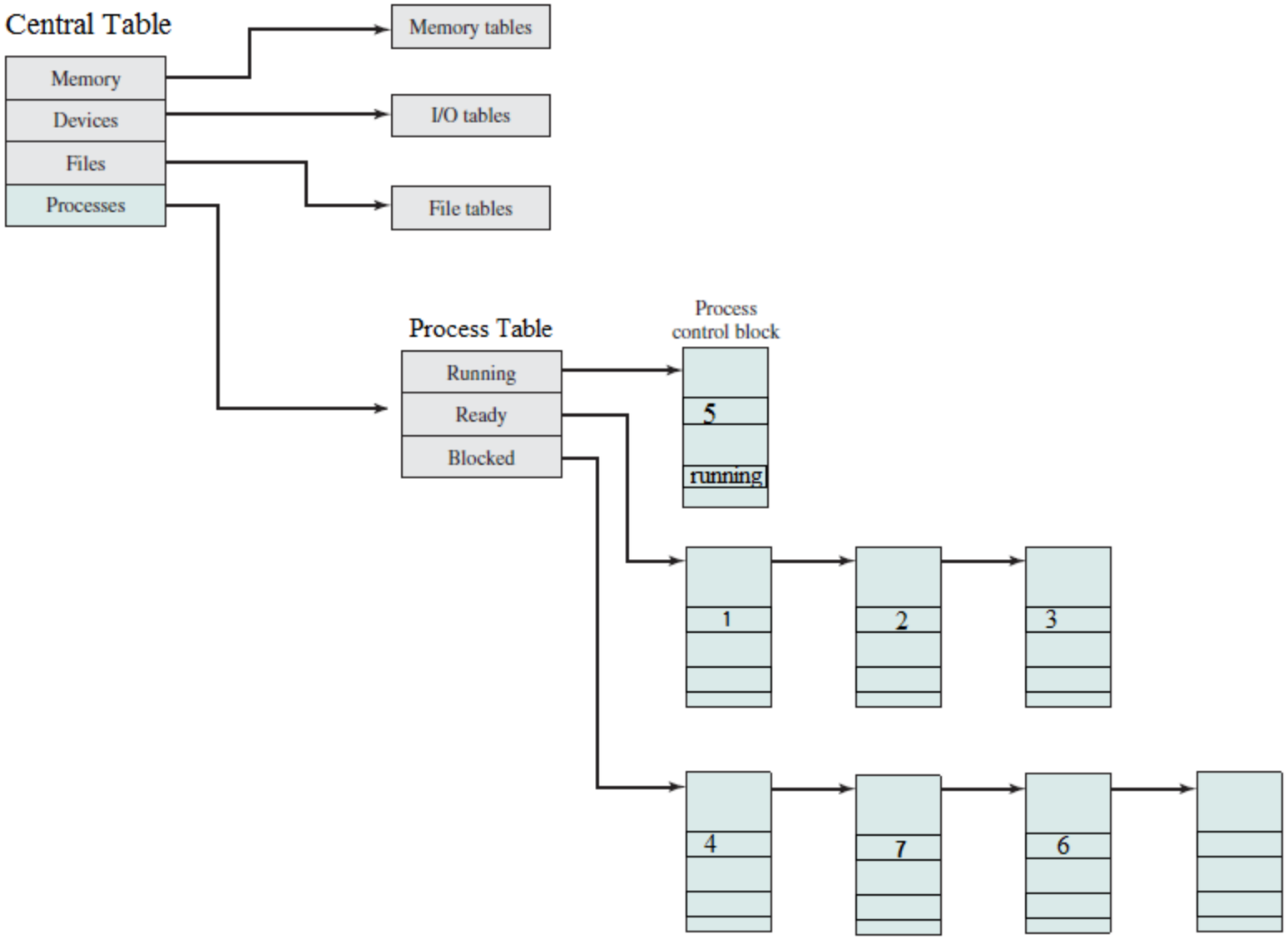
شكل رقم (2-5): إنتقال المعالج بين عمليتين (P0 و P1).

2.8 تركيبة التحكم الخاصة بنظام التشغيل Operating System Control Structures

يتحكم نظام التشغيل في كل وحدة من وحداته عن طريق الجداول:

1. **جدول الذاكرة**: تحتوي **معلومات عن الذاكرة الرئيسية والافتراضية** وهي
   - **تخصيص مكان للعمليات** في الذاكرة الرئيسية والثانوية.
   - **توفير معلومات الحماية** من الوصول الى اماكن محظورة.
   - وايضا كل ما تحتاج اليه **الذاكرة الافتراضية**.
2. **جدول اجهزة I/O** : معلومات تخص هذه الاجهزة مثل معلومات عن حالة الاجهزة.
3. **جدول الملفات** : وتشمل **اماكن** تواجدها - **حالة** الملفات - **خواصها**
4. **جدول العمليات** : وتشمل المعلومات الموجودة **في تركيبة التحكم في العمليات (PCB)**.

الشكل التالي يوضح هذه العلاقة (الشكل 2-6)

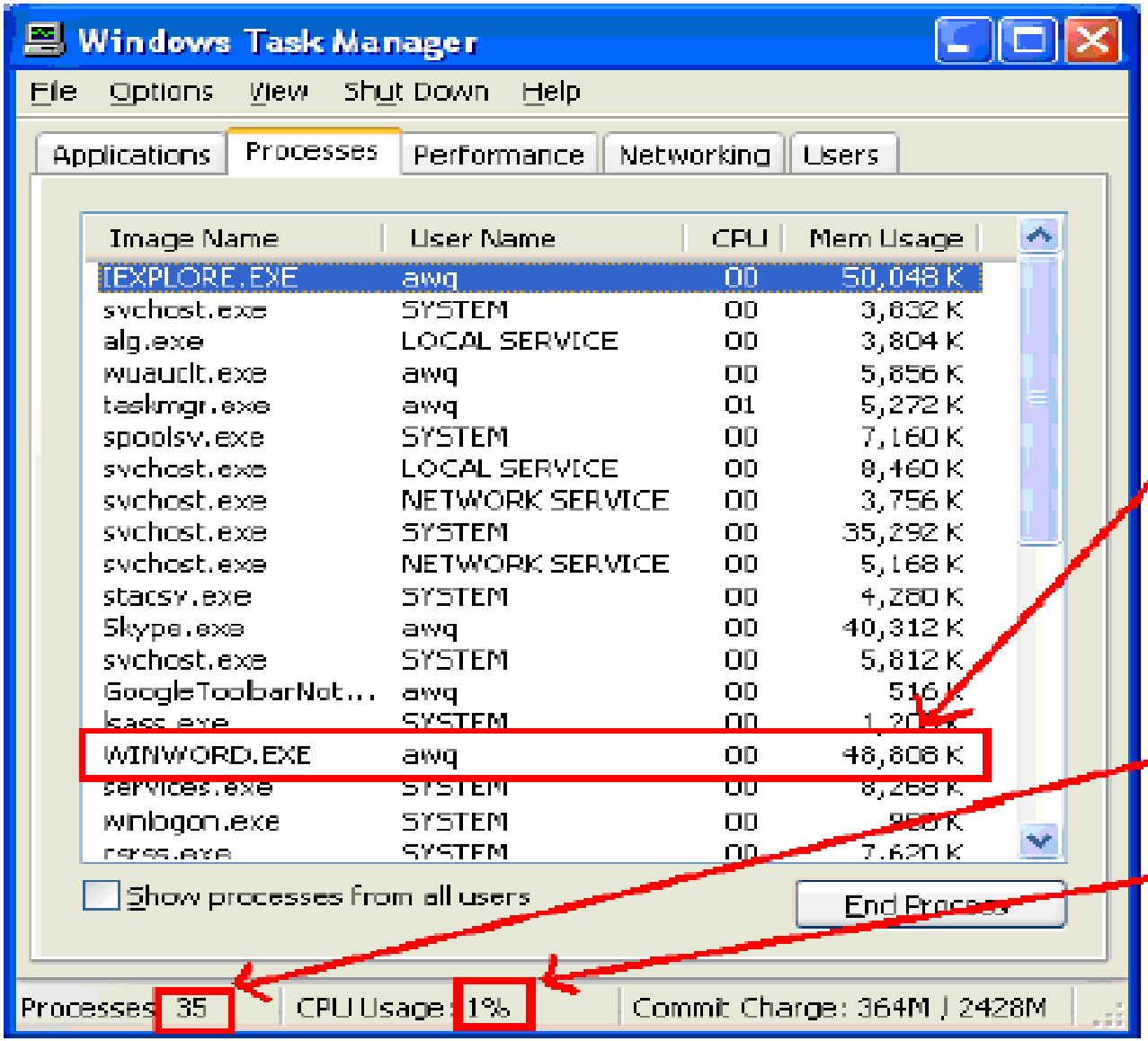


شكل رقم (2-6): كيف يتحكم نظام التشغيل في الوحدات المختلفة

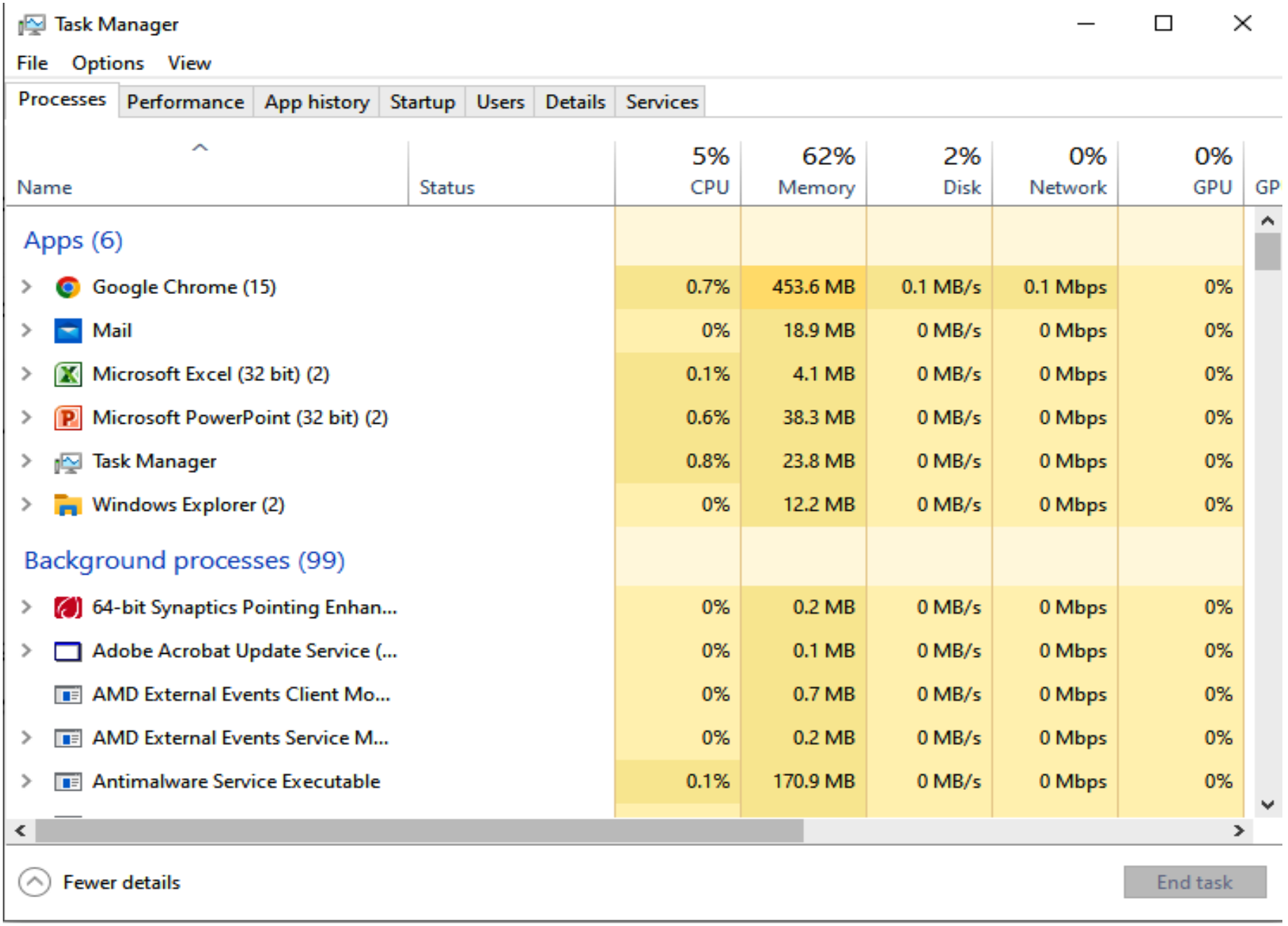
2.9 العمليات في ويندوز

عندما ننفذ برامجنا على ويندوز، لابد لنظام التشغيل من معرفة كيف يدير هذه البرامج للتأكد من أن كل برنامج أخذ حصته من الوقت في المعالج وفي الوصول إلى الذاكرة وأجهزة الدخل والخرج. ولتحقيق ذلك يتعامل نظام التشغيل مع كل برنامج كعملية. فإذا قمنا بتشغيل برنامج فسينشئ عملية لهذا البرنامج، وإذا نفذنا نسخة من نفس البرنامج فسيقوم نظام التشغيل بإنشاء عملية أخرى لهذه النسخة، بحيث تكون هنالك عملية لكل نسخة شغالة من البرنامج. وبالتالي كل البرامج التي تعمل في نظامك هي عبارة عن عمليات يدير ويتابع عملها نظام التشغيل.

لمعرفة العمليات التي تنفذ بجهازك حاليا قم بالضغط على **Ctrl+Alt+Del** فتظهر نافذة، أنقر على تبويب **Processes** فترى كل العمليات التي تعمل الآن في جهازك بما فيها عمليات نظام التشغيل ومضادات الفيروسات والبرامج الخدمية وكل برامجك المفتوحة، وترى حجم الذاكرة الذي تستخدمه كل عملية، الشكل 2-7

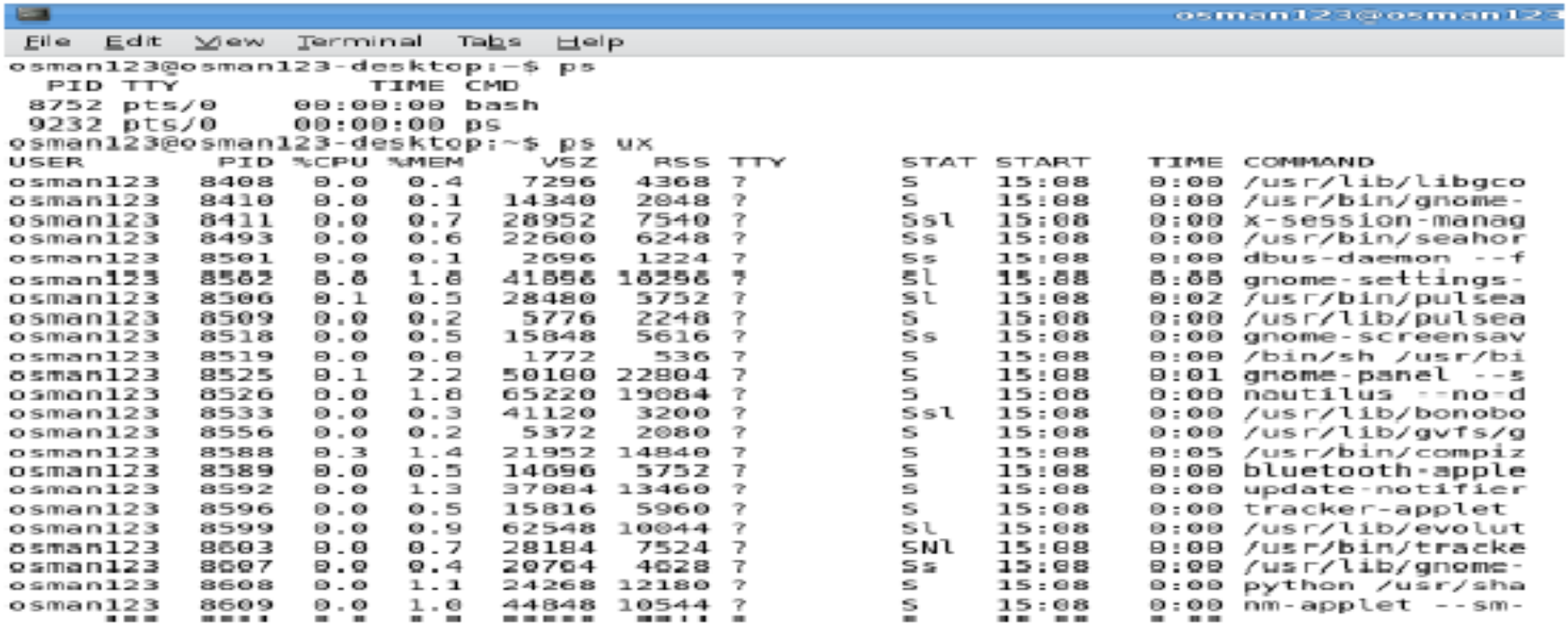


شكل رقم (2-7): مشاهدة العمليات في ويندوز.


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
Name Status 5% CPU 62% Memory 2% Disk 0% Network 0% GPU GP
Apps (6)
Google Chrome (15) 0.7% 453.6 MB 0.1 MB/s 0.1 Mbps 0%
Mail 0% 18.9 MB 0 MB/s 0 Mbps 0%
Microsoft Excel (32 bit) (2) 0.1% 4.1 MB 0 MB/s 0 Mbps 0%
Microsoft PowerPoint (32 bit) (2) 0.6% 38.3 MB 0 MB/s 0 Mbps 0%
Task Manager 0.8% 23.8 MB 0 MB/s 0 Mbps 0%
Windows Explorer (2) 0% 12.2 MB 0 MB/s 0 Mbps 0%
Background processes (99)
64-bit Synaptics Pointing Enhan... 0% 0.2 MB 0 MB/s 0 Mbps 0%
Adobe Acrobat Update Service (... 0% 0.1 MB 0 MB/s 0 Mbps 0%
AMD External Events Client Mo... 0% 0.7 MB 0 MB/s 0 Mbps 0%
AMD External Events Service M... 0% 0.2 MB 0 MB/s 0 Mbps 0%
Antimalware Service Executable 0.1% 170.9 MB 0 MB/s 0 Mbps 0%
Fewer details
End task

2.10 العمليات في لينكس (Linux)

يمكنك في **لينكس** معرفة العمليات التي تعمل الآن في جهازك، بما فيها **رقم تعريف العملية process identification number (PID)**، باستخدام الأمر . **ps** يمكننا استخدام الأمر **ps** في نظام التشغيل **أوبونتو Ubuntu**، وهو أحد توزيعات لينكس، لمشاهدة العمليات التي تعمل الآن في النظام كما في الشكل 2-7.

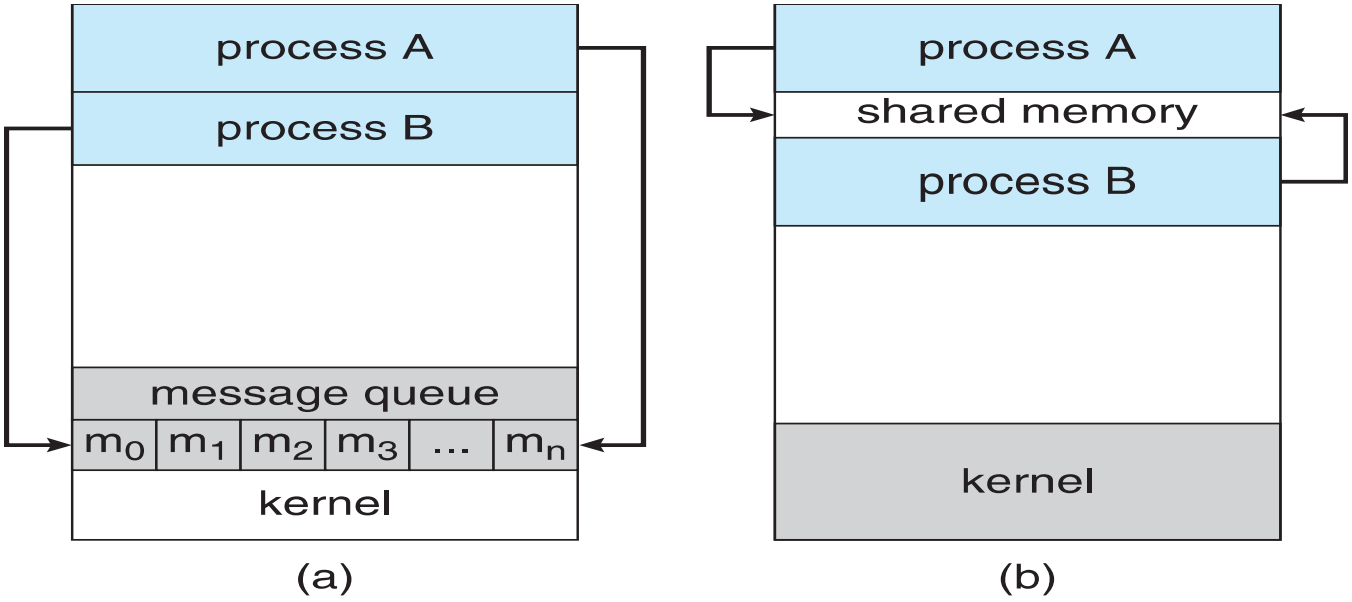


شكل رقم (2-8): مشاهدة العمليات في لينكس (أوبونتو) بالأمر: ps us.

2.11 الاتصال بين العمليات interprocess communication(IPC)

قد تكون العمليات الموجودة في النظام **مستقلة أو متعاونة** (**independent or cooperating**)
العمليات **المتعاونة** تؤثر وتتأثر بما حولها من عمليات، أما العمليات **المستقلة** فلا.
العمليات **المتعاونة** تحتاج اتصال فيما بينها (**interprocess communication(IPC**)، حيث يوجد **نوعين من طرق الاتصال**، كما موضح في **الشكل 3-9.**

- **وجود ذاكرة مشتركة** Shared memory
- **عن طريق تبادل الرسائل** Message passing



الشكل 3-9 الاتصال بين العمليات

تمارين محلولة

أختار الإجابة الصحيحة:

1. العمليات المتعاونة تتصل فيما بينها للآتي:

- لنشر المعلومات.
- لتقليل سرعة التنفيذ.
- للبعد الجغرافي.
- لا شيء مما ذكر (1)

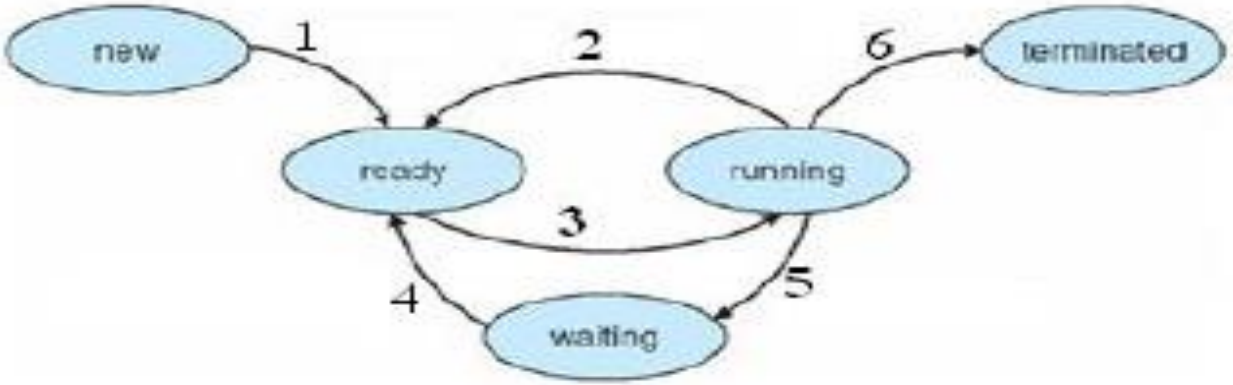
2. أجب بنعم أو لا مع تصحيح الإجابة الخاطئة:

- عندما ينتقل المعالج لتنفيذ عملية، على النظام حفظ PCB العملية القديمة وتحميل PCB العملية الجديدة. نعم
- قد يقوم الأب بإنهاء العملية الابن (abort) إذا زاد الأب في عدد الموارد المخصصة له. لا، إذا زاد الأبن
- نستخدم في لينكس استدعاء النظام (exec system call) لإنشاء عملية جديدة. لا، fork()

- عندما تنشئ عملية عملية أخرى سيكون تنفذ العملية وأبناءها بالتوالي فقط. لا،

قد يتم تنفيذ العمليات معا في وقت واحد او يمكن ان تنتظر بعضها البعض

3. وضح أسباب الانتقال بين حالات العمليات المبنية بالرسم أعلاه:



1 . دخول 2. مقاطعة 3. مجدول 4. إكتمال حدث/دخل/خرج

5. .حدث/دخل/خرج 6. خروج